

FIG. 1

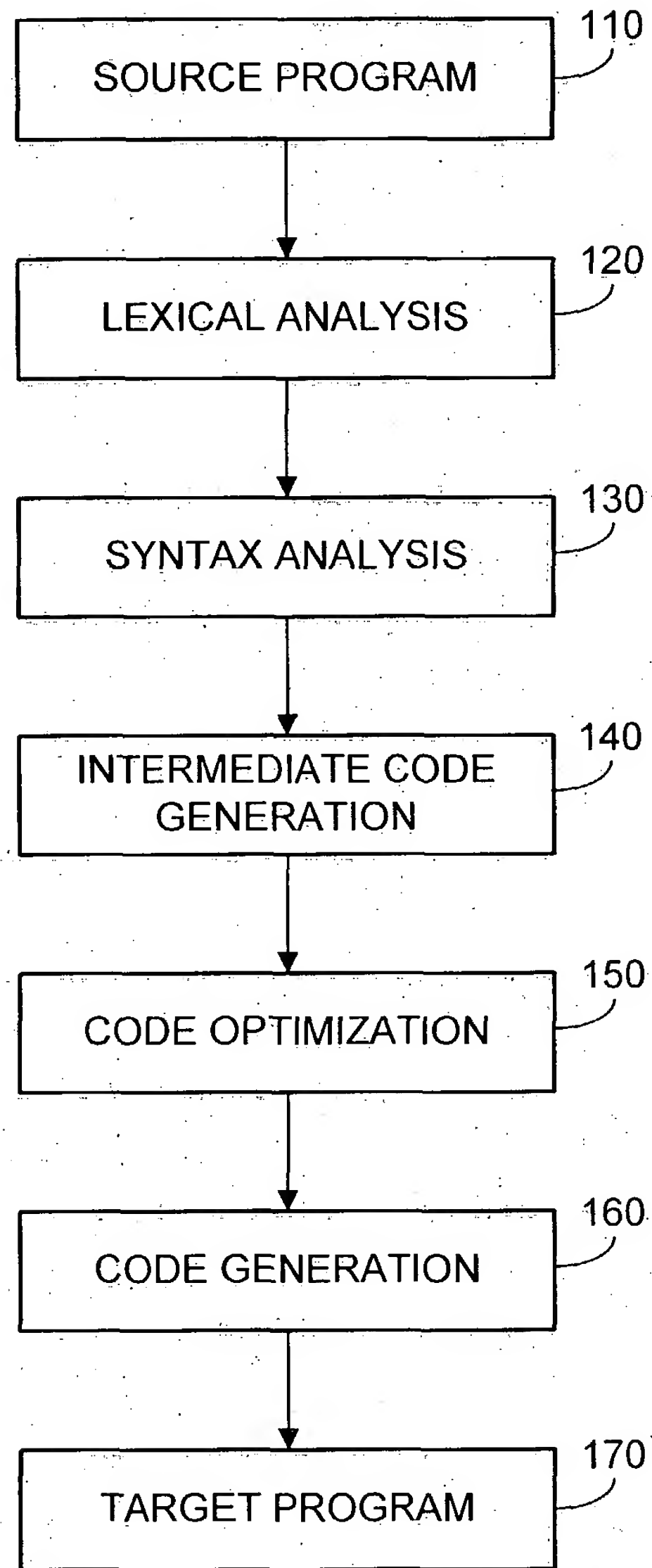


FIG. 2

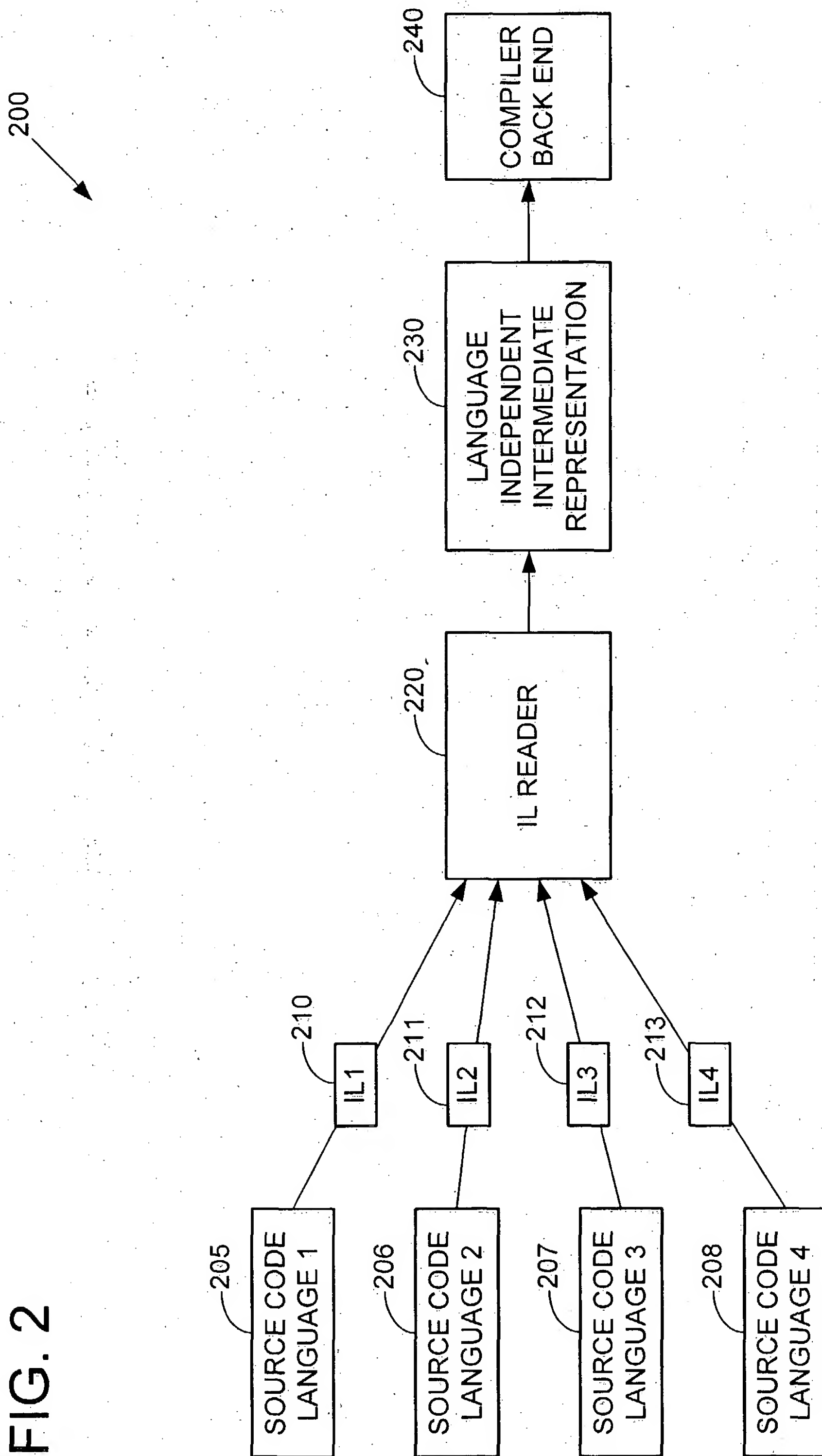


FIG. 3A

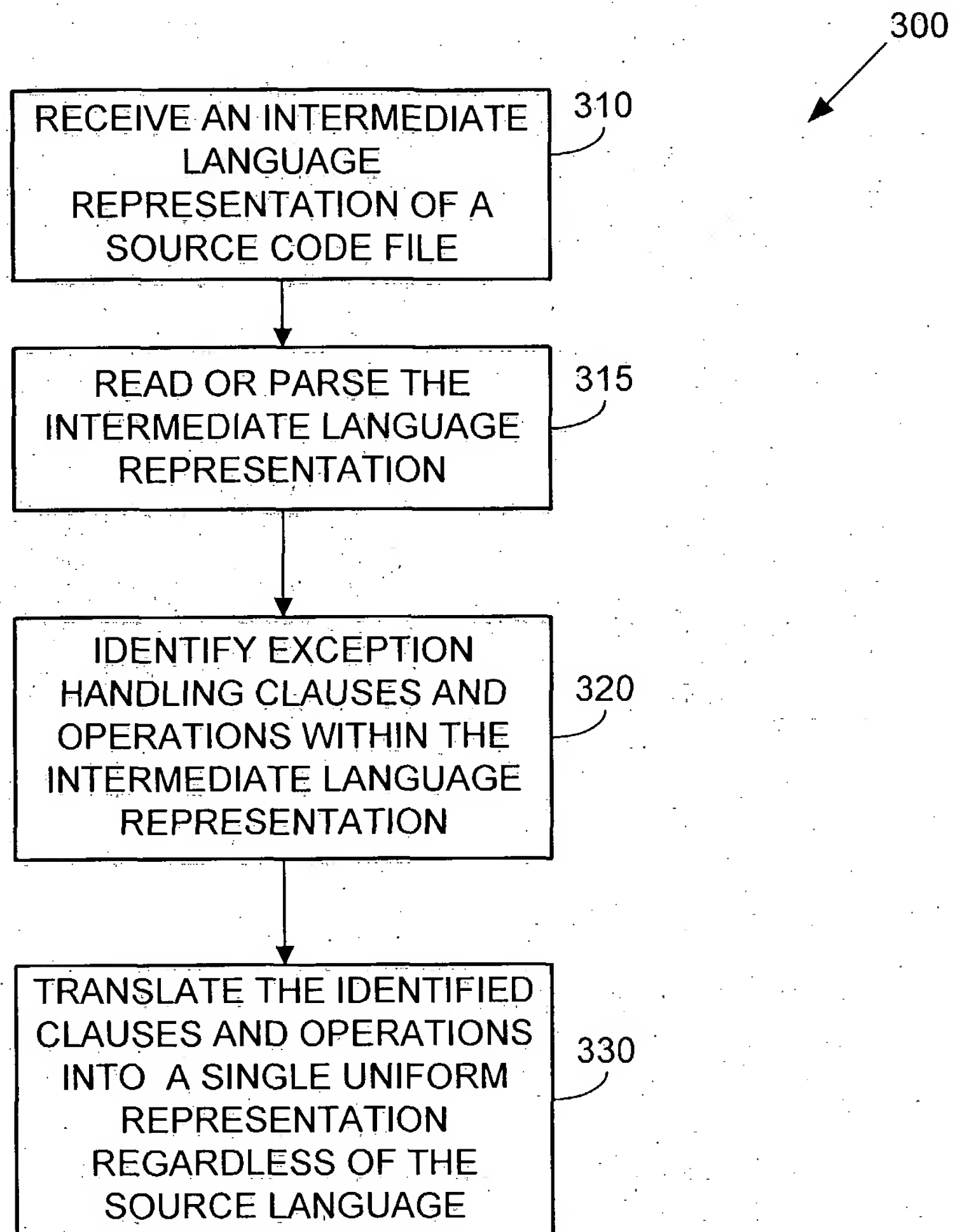


FIG. 3B

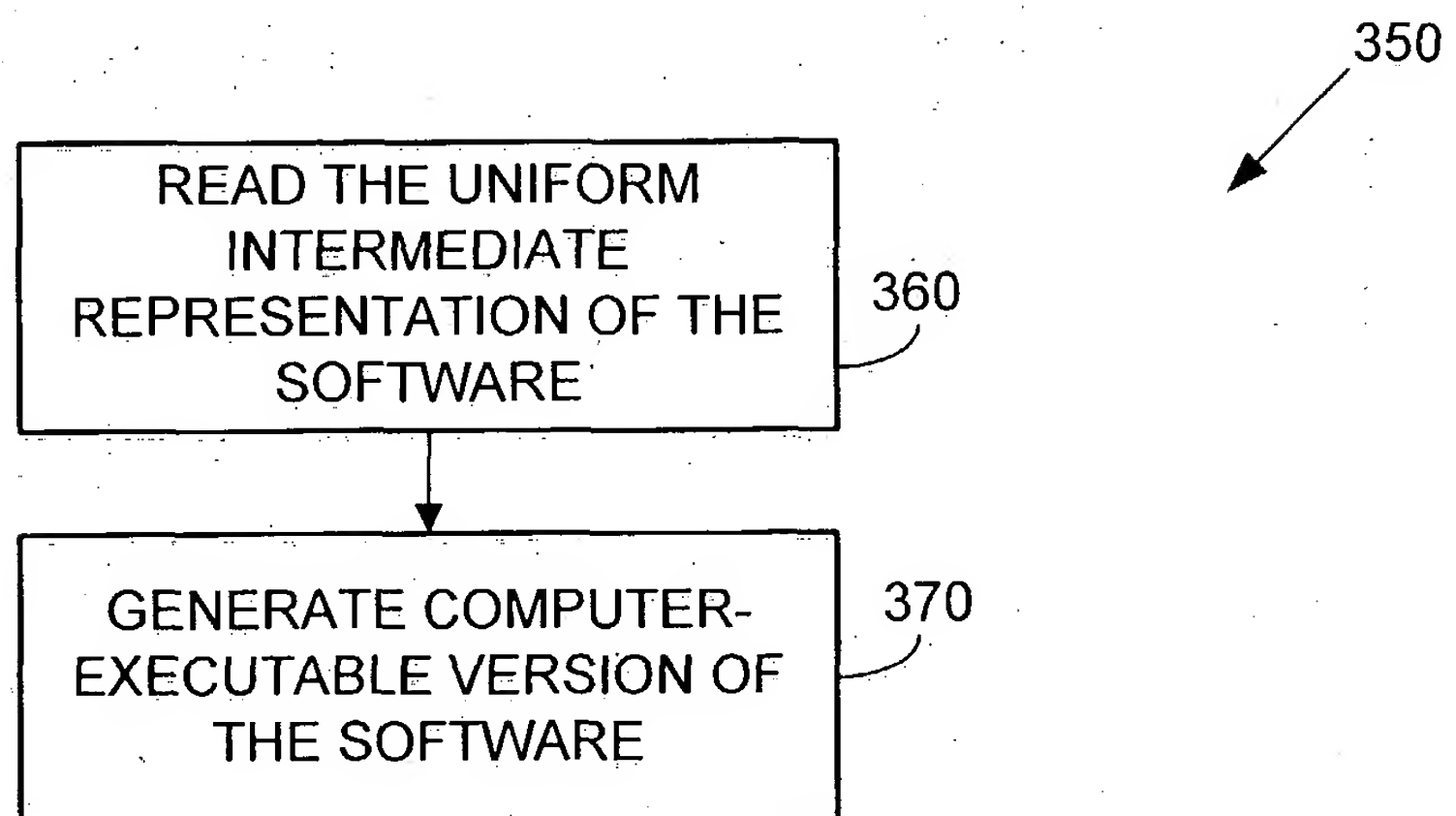


FIG. 4

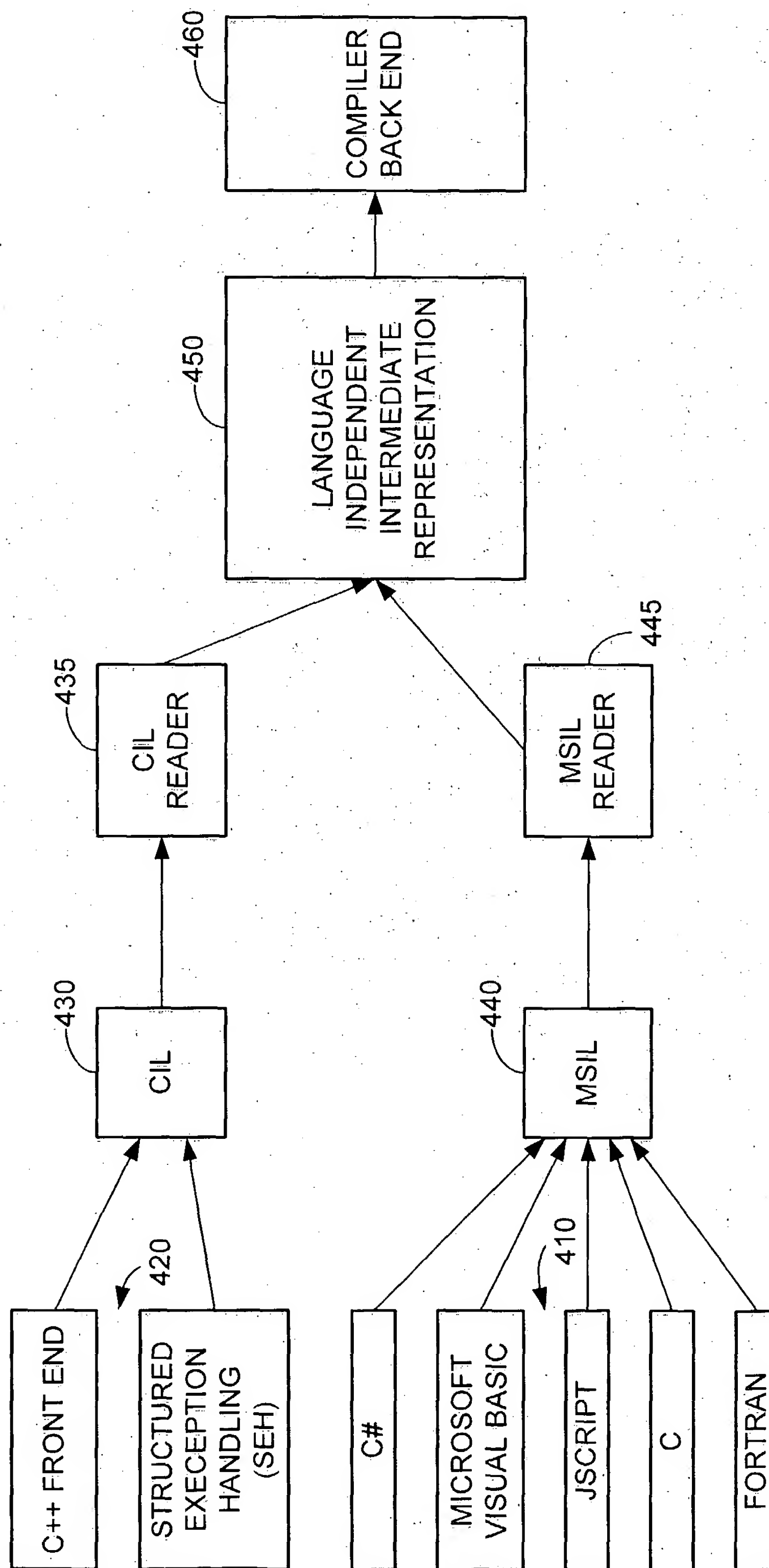


FIG. 5

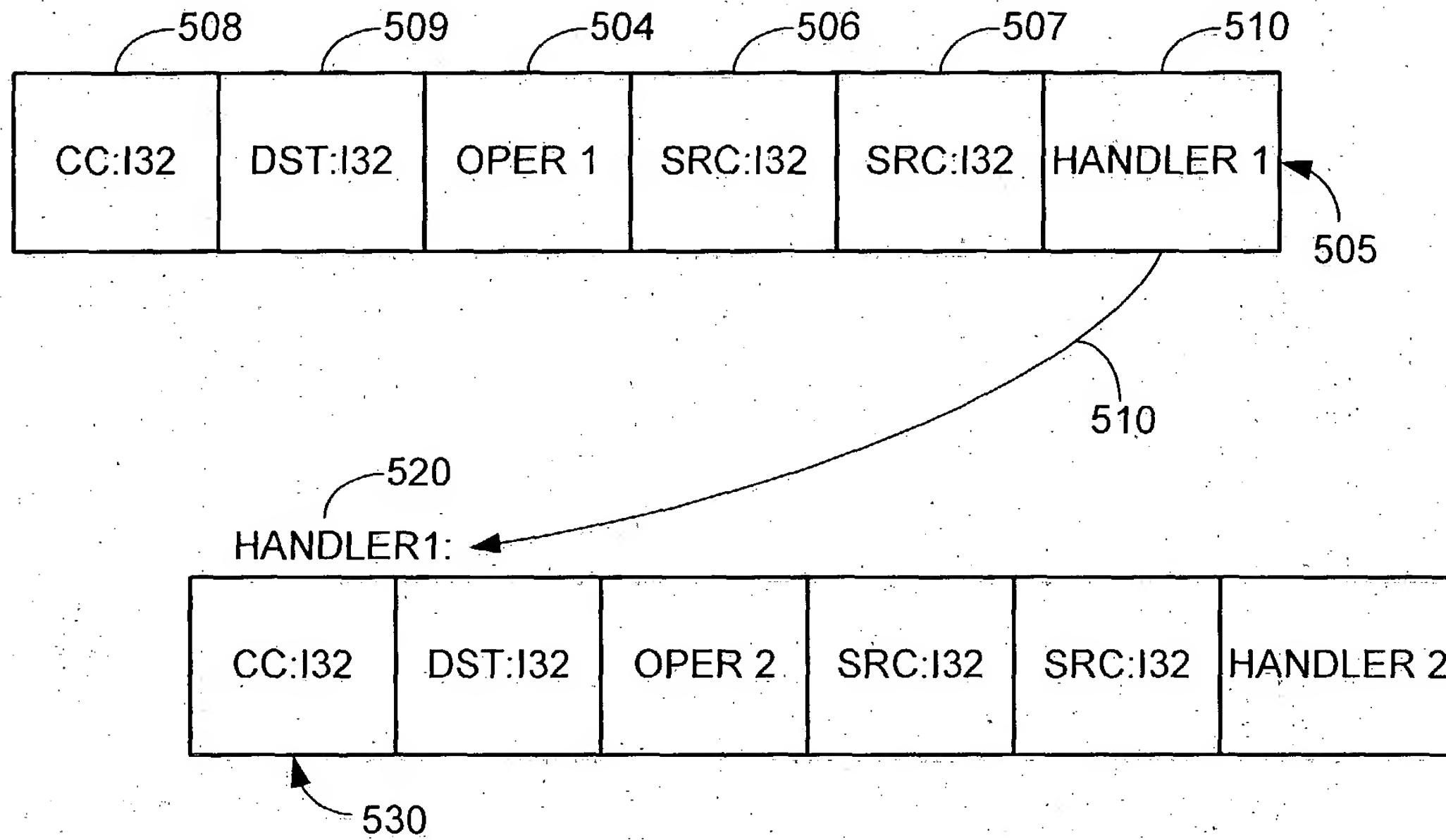


FIG. 6

```
void foo(int a, int b, int c, int d)
{
    x = a div b;
    x = c div d;
}
```

FIG. 7

```
a.int32, b.int32, c.int32 d.int32 = ENTER foo
x.int32 = DIV a.int32, b.int32; $HANDLER
x.int32 = DIV c.int32, d.int32; $HANDLER 710
EXIT
$HANDLER:
UNWIND
EXIT
```

FIG. 8

```
void foo(int a, int b, int c, int d)
{
    try
    {
        x = a + b;
        x = x + c * d;
    }
    finally
    {
        x = x + 1;
    }
}
```

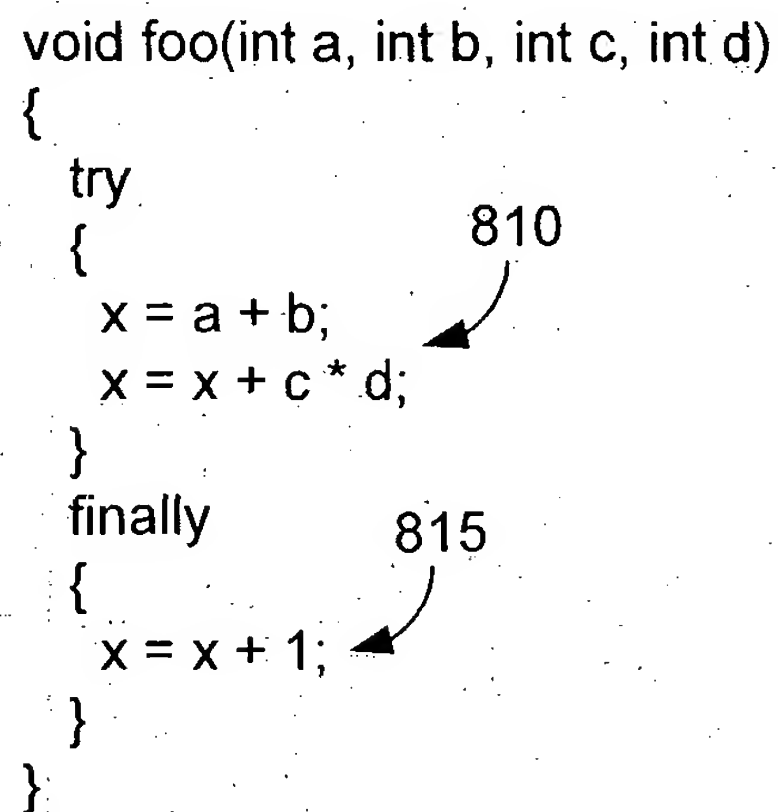


Diagram illustrating a try-finally block structure. The try block contains two lines of code: `x = a + b;` and `x = x + c * d;`. The finally block contains one line of code: `x = x + 1;`. Arrows point from labels 810 and 815 to the corresponding code lines.

FIG. 9

```
a.int32, b.int32, c.int32 d.int32 = ENTER foo
x.int32 = ADD a.int32, b.int32;
t.int32 = MUL c.int32, d.int32;
x.int32 = ADD x.int32, t.int32;
FINAL $FINALIZE, $END
$FINALIZE:
    e.obj32, r.code = FINALLY;
    x.int32 = ADD x.int32, 1.int32;
    ENDFINALLY e.obj32, r.code, $END;
$END:
    EXIT;
```

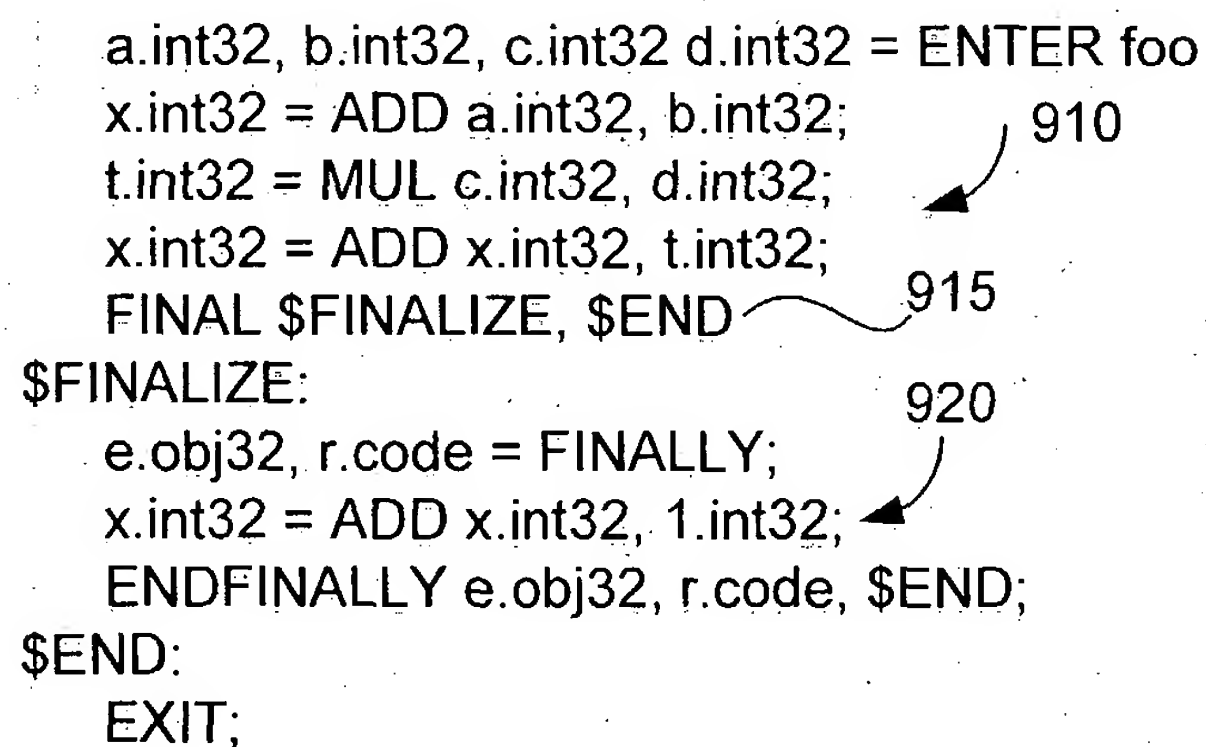


Diagram illustrating a sequence of code blocks. The code starts with a line `a.int32, b.int32, c.int32 d.int32 = ENTER foo`. This is followed by three lines of code: `x.int32 = ADD a.int32, b.int32;`, `t.int32 = MUL c.int32, d.int32;`, and `x.int32 = ADD x.int32, t.int32;`. These three lines are grouped by a bracket labeled 910. The next line is `FINAL $FINALIZE, $END`, which is labeled 915. This is followed by a block of code labeled 920, which contains three lines: `$FINALIZE:`, `e.obj32, r.code = FINALLY;`, `x.int32 = ADD x.int32, 1.int32;`, and `ENDFINALLY e.obj32, r.code, $END;`. The final line is `$END:` followed by `EXIT;`.

FIG. 10

```

void foo(int a, int b, int c, int d)
{
    try
    {
        x = a div b;
        x = c div d;
    }
    finally
    {
        x = x + 1;
    }
}
    
```

Diagram annotations for FIG. 10:

- A bracket on the right side of the try block (lines 5-6) is labeled 1010.
- An arrow points from the label 1015 to the line `x = x + 1;` in the finally block (line 10).

FIG. 11

```

a.int32, b.int32, c.int32 d.int32 = ENTER foo
x.int32 = DIV a.int32, b.int32; $FINALIZE
x.int32 = DIV c.int32, d.int32; $FINALIZE
FINAL $FINALIZE, $END
$FINALIZE:
    e.obj32, r.code = FINALLY;
    x.int32 = ADD x.int32, 1.int32;
    ENDFINALLY e.obj32, r.code, $END; $PROPAGATE
$END:
    EXIT;
$PROPAGATE:
    UNWIND
    EXIT;
    
```

Diagram annotations for FIG. 11:

- A bracket on the right side of the two \$FINALIZE lines is labeled 1110.
- An arrow points from the label 1115 to the line `e.obj32, r.code = FINALLY;` in the \$FINALIZE block.
- An arrow points from the label 1120 to the line `ENDFINALLY e.obj32, r.code, $END; $PROPAGATE`.



FIG. 12

1240

```
1 void foo(int a, int b, int c, int d)
2 {
3   try
4   {
5     x = a div b;
6     x = c div d;
7   }
8   catch (System.DivideByZeroException f) ~1215
9   {
10    b = 1;
11    d = 1;
12  }
13  catch (System.Exception e) ~1225
14  {
15    bar();
16  }
17 }
```

1210

1220

1231

FIG. 13

```
a.int32, b.int32, c.int32 d.int32 = ENTER foo
x.int32 = a.int32 DIV b.int32; $HANDLER1
x.int32 = c.int32 DIV d.int32; $HANDLER1 > 1310
GOTO $END$
$HANDLER1:
  F.DivideByZeroException = TYPEFILTER $CATCH1,
HANDLER2;
$CATCH1:
  b.int32 = ASSIGN 1.int32;
  d.int32 = ASSIGN 1.int32;
  GOTO $END
$HANDLER2:
  E.Exception = TYPEFILTER $CATCH2, $PROPAGATE;
$CATCH2:
  CALL bar(); $PROPAGATE
  GOTO $END
$PROPAGATE:
  UNWIND ~1335
EXIT;
```

1310

1315

1320

1325

1330

1335

FIG. 14

```
void foo(int a, int b, int c, int d)
{
    try
    {
        x = a div b; 1410
        x = c div d;
    }
    catch (System.DivideByZeroException f) 1415
    {
        b = 1; 1420
        d = 1;
    }
    catch (System.Exception e) 1425
    {
        bar(); 1430
    }
    finally 1440
    {
        x = x + 1;
    }
}
```

## FIG. 15

```
a.int32, b.int32, c.int32 d.int32 = ENTER foo
  x.int32 = a.int32 DIV b.int32; $HANDLER1
  x.int32 = c.int32 DIV d.int32; $HANDLER1
  FINAL $FINALIZE, $END
$HANDLER1:
  F.DivideByZeroException = TYPEFILTER $CATCH1,
$HANDLER2;
$CATCH1:
  b.int32 = ASSIGN 1.int32;
  d.int32 = ASSIGN 1.int32;
  FINAL $FINALIZE, $END;
$HANDLER2:
  E.Exception = TYPEFILTER $CATCH2, $FINALIZE;
$CATCH2:
  CALL bar(); $FINALIZE
  FINAL $FINALIZE, $END;
$FINALIZE:
  e.obj32, r.code = FINALLY;
  x.int32 = ADD x.int32, 1.int32;
  ENDFINALLY e.obj32, r.code, $END; $PROPAGATE
$PROPAGATE:
  UNWIND
  EXIT;
$END:
  EXIT;
```

1510

1520

1521

1525

1530

1531

1540

1550

1560

FIG. 16

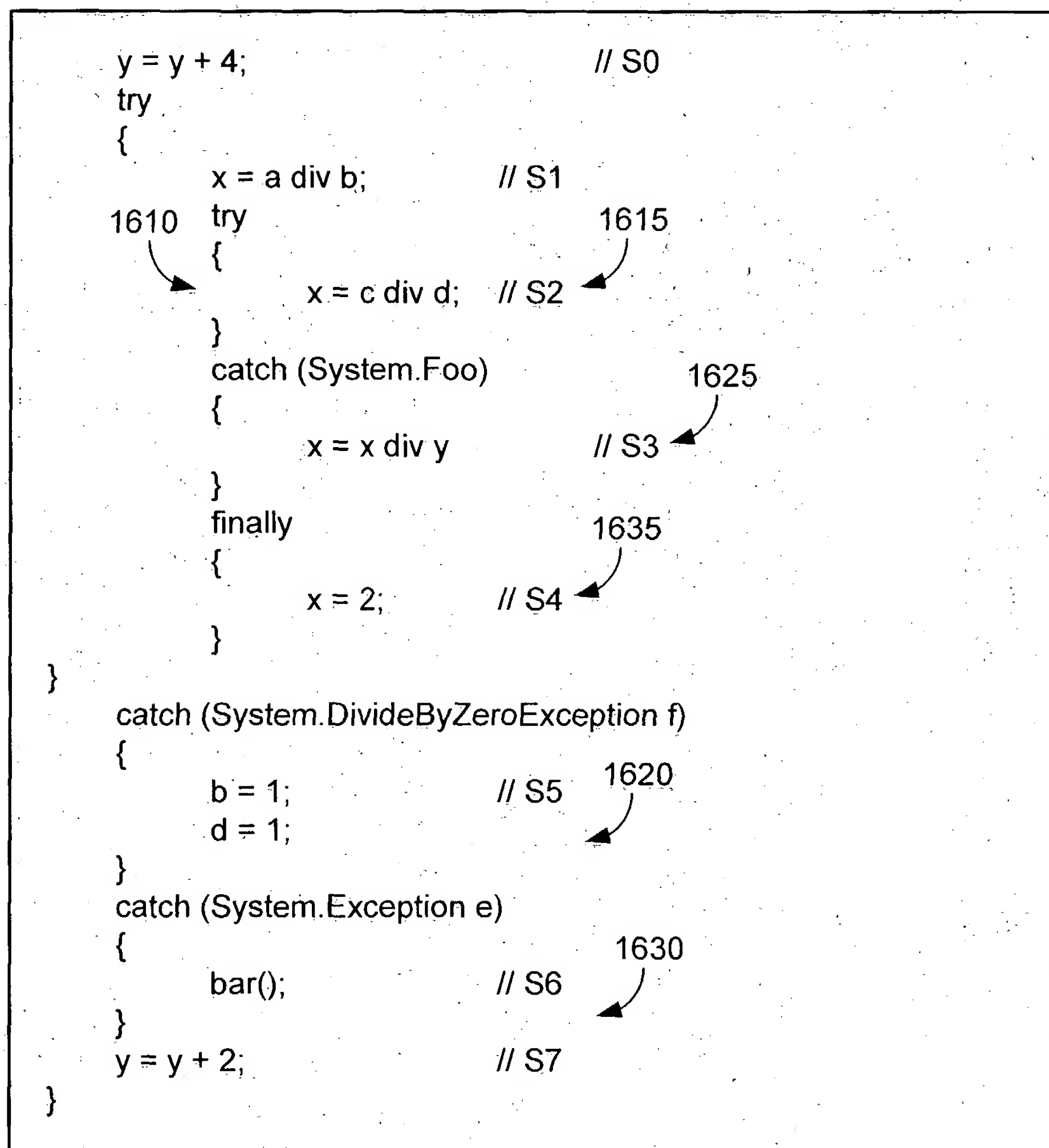




FIG. 18

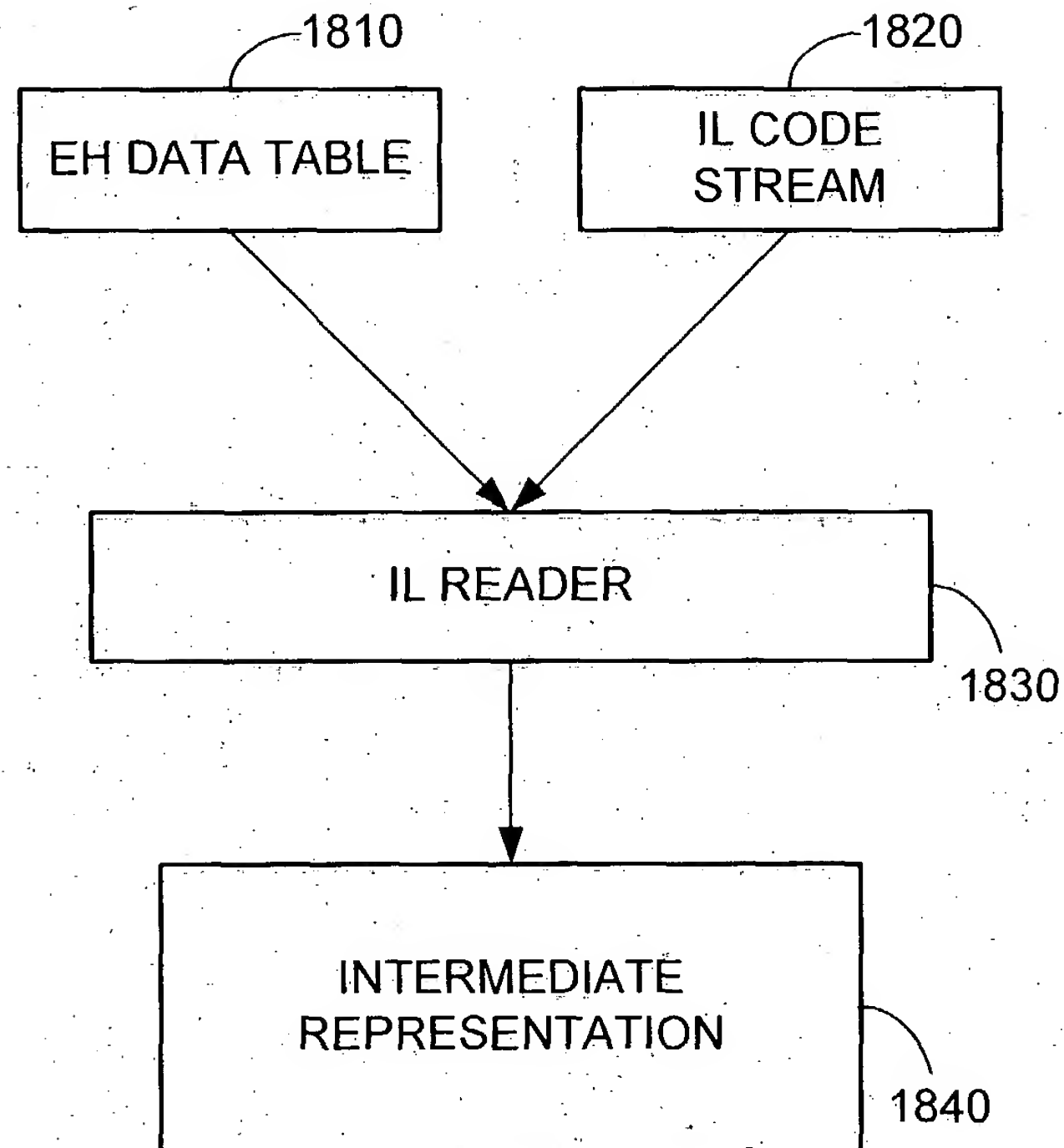


FIG. 19A

ENTRY	INFO TAG	PROTECTED BLOCK	DESTINATION/HANDLER BLOCK
1	TRY CATCH	3-7	8-12
2	TRY CATCH	3-7	13-16

FIG. 19B

DESTINATION/HANDLER BLOCK OFFSET	LABEL
8-12	HANDLER 1
13-16	HANDLER 2

FIG. 19C

PROTECTED BLOCK OFFSET	DESTINATION/HANDLER BLOCK OFFSET	LABEL
3-7	8-12	HANDLER 1
3-7	13-16	HANDLER 2

FIG. 20

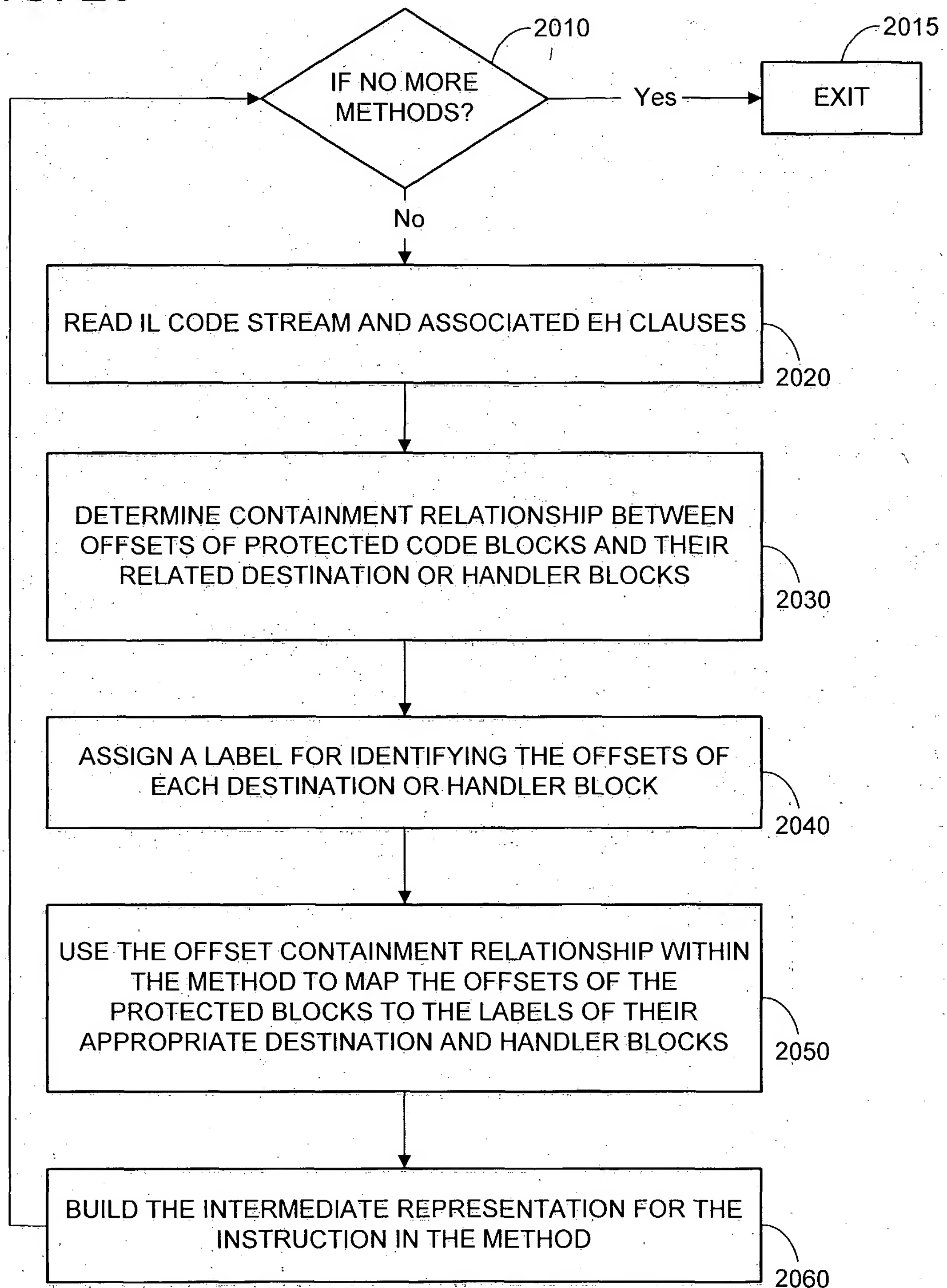




FIG. 21

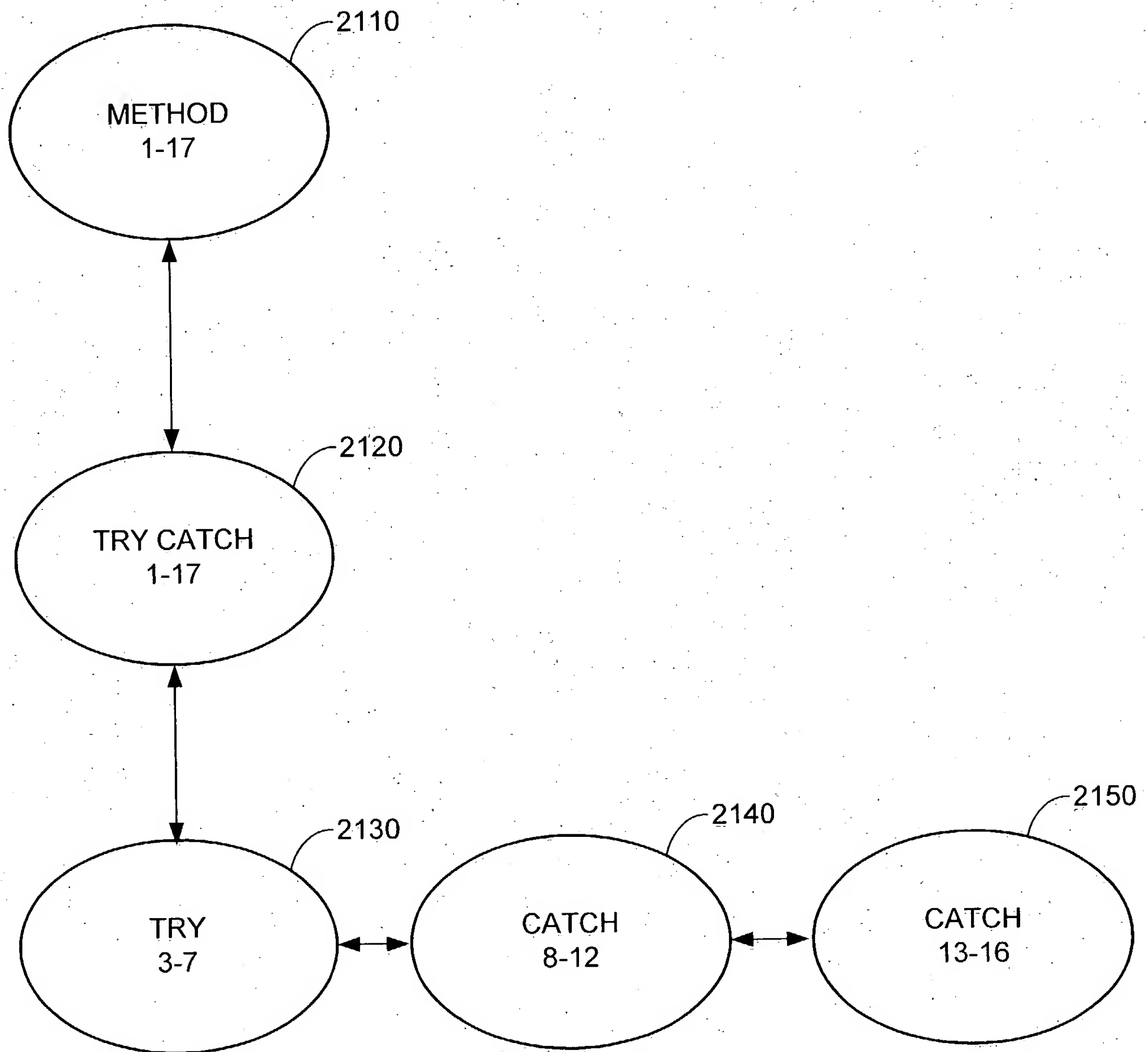


FIG. 22

```
void proc()
{
    class1 obj1; // S1
    obj1.foo(); // S2
    class2 obj2; // S3
    obj2.bar(); // S4
}
```

FIG. 23

```
void proc()
{
    ctor1(&obj1); ← 2310
    try
    {
        obj1.foo();
        ctor2(&obj2);
        try
        {
            obj2.bar(); ← 2330
        }
        finally
        {
            dtor2(&obj2); ← 2340
        }
    }
    finally
    {
        dtor1(&obj1) ← 2320
    }
}
```

## FIG. 24

```
ENTER proc
CALL class1, &_obj1 $PROPAGATE
CALL foo, &_obj1 $DTOR1
CALL class1, &_obj2 $DTOR1
CALL bar, &_obj2 $DTOR2
FINAL $DTOR2, $NEXT; ~2410
$NEXT:
    FINAL $DTOR1, $END ~2420
$DTOR2:
    e, r = FINALLY
    CALL DTOR2(&obj2); $DTOR1 ~2435
    ENDFINALLY e, r, $DTOR1, $NEXT
$DTOR1:
    e2, r2 = FINALLY
    CALL DTOR1(&obj1); $PROPAGATE ~2445
    ENDFINALLY e1, r2, $PROPAGATE, $END
$PROPAGATE:
    UNWIND
    EXIT;
$END:
    EXIT;
```

Diagram annotations: An arrow points from label 2430 to the line "CALL DTOR2(&obj2); \$DTOR1 ~2435". Another arrow points from label 2440 to the line "CALL DTOR1(&obj1); \$PROPAGATE ~2445".

## FIG. 25

```
void proc(int x)
{
    foo(x ? obj1(x) : obj2(x+1));
}
```

## FIG. 26

```
void proc()
{
    try
    {
        t1 = x ? ctor(&obj1,x) : NULL; ~2610
        try
        {
            t2 = x ? NULL : ctor(&obj2,x+1) ~2620
            foo( x ? t1 : t2);
        }
        finally
        {
            if (x) dtor(&obj1); ~2630
        }
    }
    finally
    {
        if (!x) dtor(&obj2); ~2640
    }
}
```

FIG. 27

<u>x</u>	=	ENTER proc	
t140	=	CMP(NE) <u>x</u> , 0	
		CBRANCH(NE) t140, \$L4, \$L5 ~ 2710	
\$L4:			
2720 t134	=	CALL ctor, &obj1, <u>x</u> ; \$PROPAGATE	2721
t135	=	ASSIGN t134	
tv141-	=	ASSIGN [t135]	
\$t142	=	ASSIGN 1	
		GOTO \$L6	
\$L5:			
t137	=	ADD <u>x</u> , 1	2731
2730 t138	=	CALL ctor, &obj2, t137; \$PROPAGATE	
t139	=	ASSIGN t138	
tv141-	=	ASSIGN [t139]	
\$t142	=	ASSIGN 0	
		GOTO \$L6	
\$L6:			
t145	=	ASSIGN tv141-	
		CALL bar, t145	
		FINAL \$OBJ1, \$L11	
\$L11:			
		FINAL \$OBJ2, \$L12	
\$OBJ1:			
r1	=	→ FINALLY	
t144	=	CMP(EQ) \$t142, 0	
		CBRANCH(EQ) t144, \$L9, \$L10	
\$L9:			
		CALL dtor, &obj1 \$PROPAGATE	
		GOTO \$L10	
\$L10:	2740	→ ENDFINALLY; r1, [\$L11], \$PROPAGATE	
\$OBJ2:			
r2	=	→ FINALLY	
t143	=	CMP(EQ) \$t142, 1	
		CBRANCH(EQ) t143, \$L7, \$L8	
\$L7:			
		CALL dtor, &obj2 \$PROPAGATE	
		GOTO \$L8	
\$L8:	2750	→ ENDFINALLY; r2, [\$L12], \$PROPAGATE	
\$PROPAGATE:			
		UNWIND	
		EXIT	
\$L12:			
		EXIT	

## FIG. 28

```
Obj foo(int x)
{
    Obj a , b; ~2810
    bar();
    if (x == 0)
    {
        return Obj(1); ~2820
    }
    else
    {
        return Obj(2); ~2830
    }
}
```

## FIG. 29A

```
Obj foo(int x)
{
    CALL ctor (&a); $unwind;
    CALL ctor (&b); $final_a;
    bar(); $final_b;
    if (x == 0)
    {
        CALL ctor Obj (&r1, 1); $final_b; ~2960
        $f1 = 1; ~2950
        FINAL $final_b1; L2
        L2:
        FINAL $final_a1, L1
        L1:
        $f1 = 0;
        FINAL $final_r1, Lret
        Lret:
        return r1; ~2910
    }
    CALL ctor Obj(&r2, 2); $final_b;
    $f2 = 1;
    FINAL $final_b2, L3
    L3:
    FINAL $final_a2, L4
    L4:
    $f2 = 0;
    FINAL $final_r2, Lret2
    Lret2:
    return r2;
```

## FIG. 29B

```
$final_b:
    e, R = FINALLY
    DTOR (&b); $final_a;
    ENDFINALLY e, R, $final_a;
$final_a:
    e, R = FINALLY
    CALL DTOR (&a); $unwind;
    ENDFINALLY e, R, HANDLER:$unwind;
$final_b1:
    e, R = FINALLY
    CALL DTOR (&b); $final_a1; ~2930
    ENDFINALLY e, R, [L2], $final_a1;
$final_a1:
    e, R = FINALLY
    CALL DTOR (&a); $final_r1; ~2920
    ENDFINALLY e, R, [L1]; $final_r1;
$final_b2:
    e, R = FINALLY
    CALL DTOR (&b); $final_a2;
    ENDFINALLY e, R, [L3]; $final_a2;
$final_a2:
    e, R = FINALLY
    CALL DTOR (&a); $final_r2;
    ENDFINALLY e, R, [L4]; $final_r2;
$final_r1:
    e, R = FINALLY
    if ($f1 == 1) CALL DTOR (&r1); $unwind; ~2940
    ENDFINALLY e, R, [Lret1]; $unwind;
$final_r2:
    e, R = FINALLY
    if ($f2 == 1) CALL DTOR (&r2); $unwind;
    ENDFINALLY e, R, [Lret2]; $unwind;
}
```



FIG. 30

```
void proc()
{
    class1 obj1; // S1 - create an obj of type Class1
    obj1.foo(); // S2 - calling a method on obj.1
    throw foo; // S3
}
```

FIG. 31

```
void proc()
{
    class1 obj1; // S1
    class1 temp;
    try {
        obj1.foo(); 3110 // S2
        temp.copy_ctor(obj1); //
        special_throw(&temp, &dtor_of_class1) // S3
    } 3120
    finally {
        dtor_of_class1(obj1); 3130
    }
}
```

FIG. 32

```
ENTER proc
CALL ctor1(&obj1); $PROPAGATE
CALL foo(&obj1); $DTOR1
CALL copy_ctor(&temp, &obj1); $DTOR1
FINAL $DTOR1;
THROWVAL &temp, &dtor_of_class1; $PROPAGATE
$DTOR1: 3220 3230 3210
    e2, r2 = FINALLY
    CALL DTOR1(&obj1); $PROPAGATE
    ENDFINALLY e1, r2, $PROPAGATE, $END
$PROPAGATE:
    UNWIND
    EXIT;
$END:
    EXIT;
```

FIG. 33

```
void proc()
{
    __try {
        foo();
    } __except(filter()) {
        body();
    }
    next();
}
```

## FIG. 34

```
ENTER proc
$LABEL:
  SEHENTER; $HANDLER ~ 3410
  CALL foo(); $HANDLER ~ 3420
  GOTO $NEXT;
$HANDLER: ~ 3430
  x = FILTER
  t = CALL filter(); ~ 3450
  ENDRESUMEFILTER t, $HANDLERBODY, $END, $LABEL
$HANDLERBODY: ~ 3440
  CALL body(); $PROPAGATE
  GOTO $NEXT;
$NEXT:
  CALL next();
  EXIT;
$END:
  EXIT;
```

FIG. 35

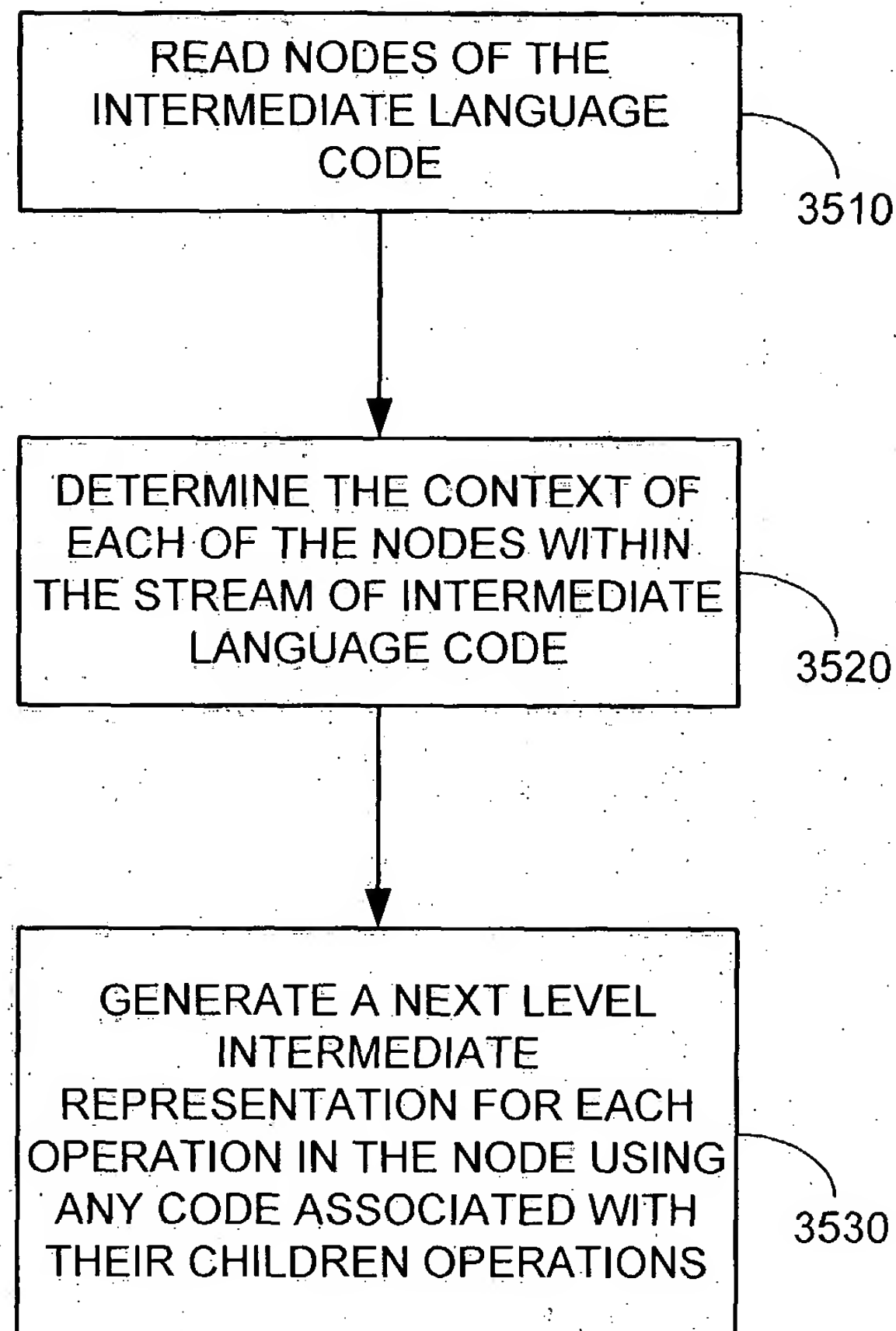


FIG. 36

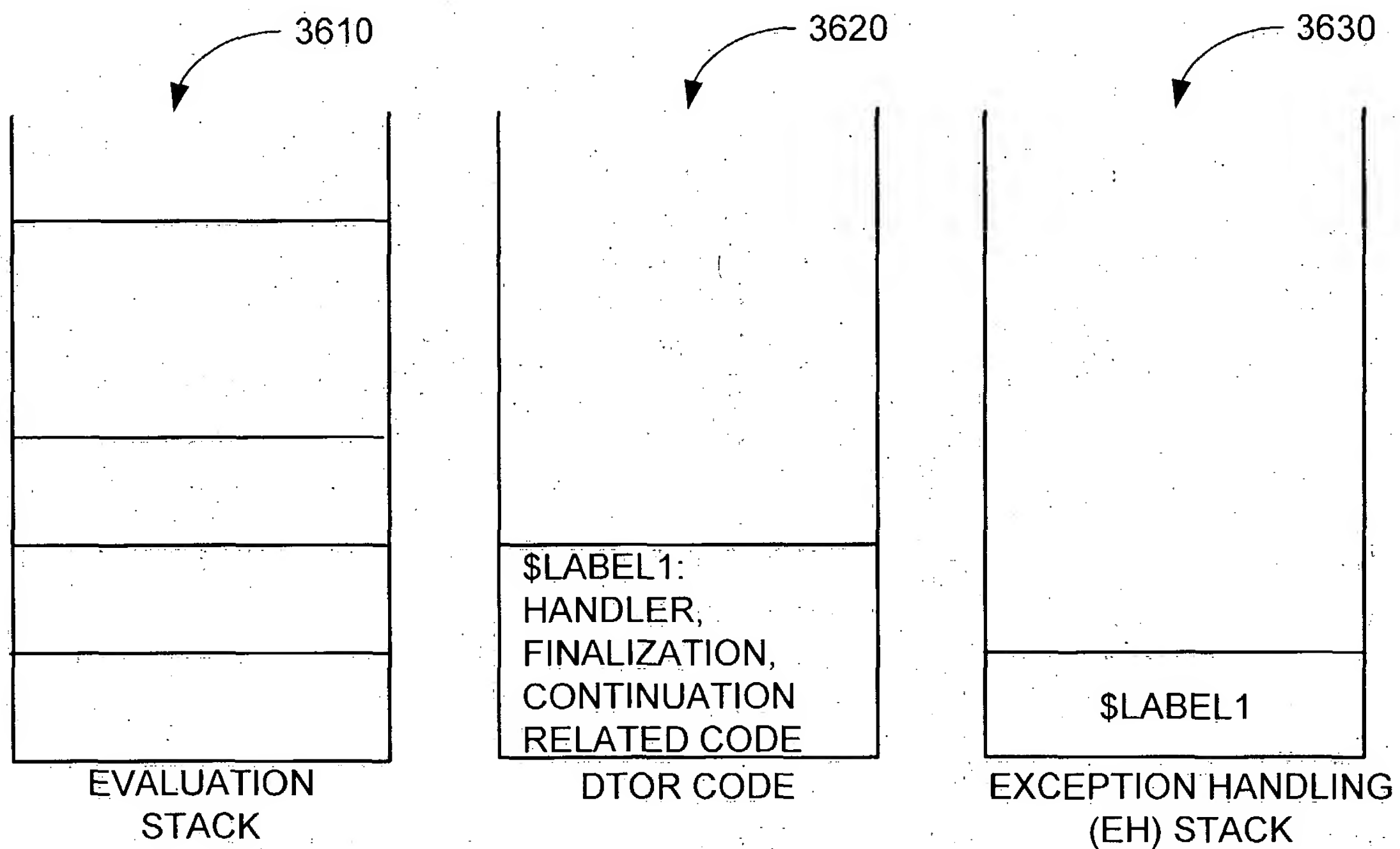
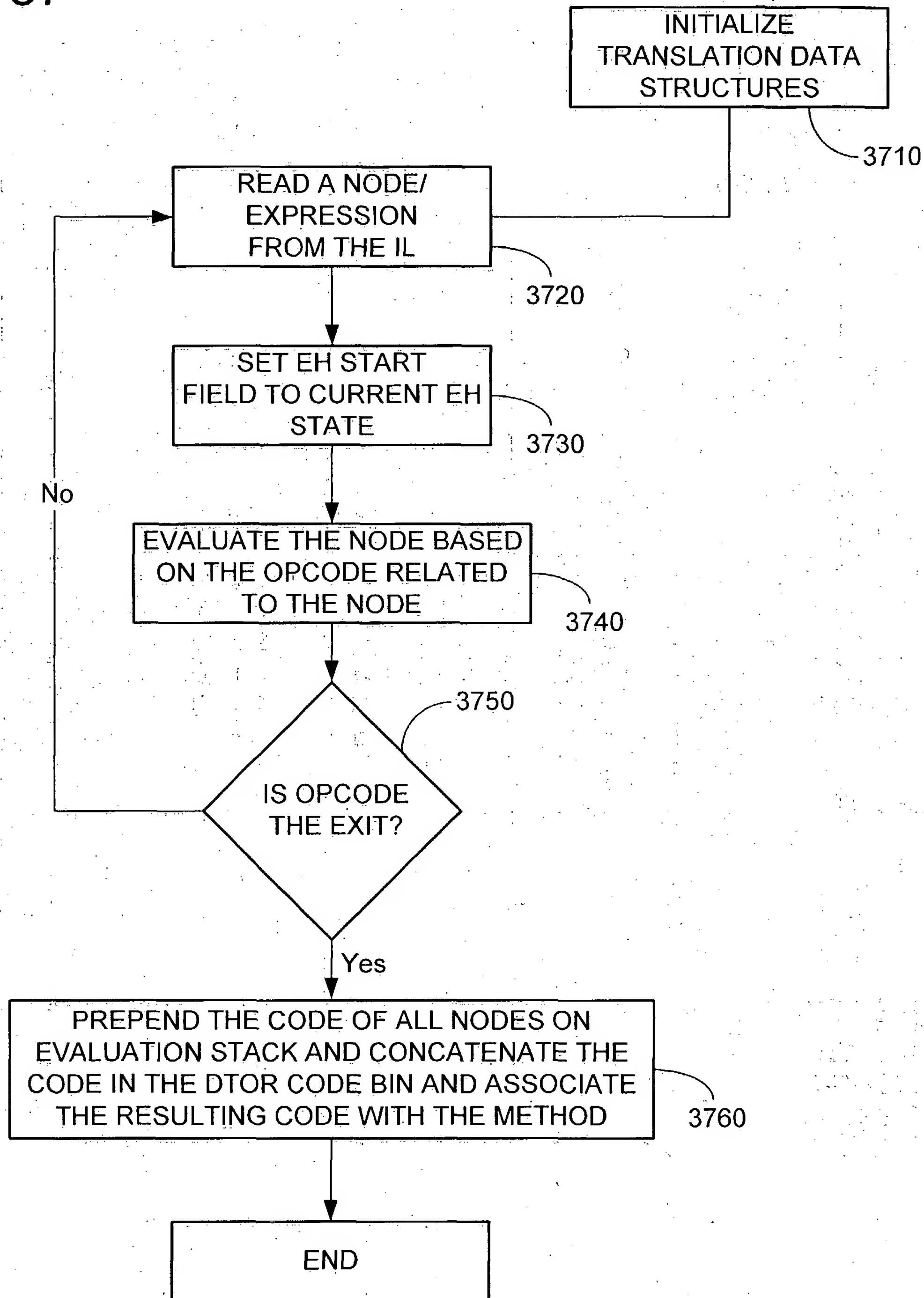


FIG. 37



## FIG. 38A

```
>>> IL for function ?proc@@YAXXZ:
OPpragma pragma(31)    PR_PRAGSTAT Pragma Status: 0x00800000
OPpragma pragma(32)    PR_INLINE Inline Status: 0x0010
OPpragma pragma(2)     PR_FILENAME Filename key(0x20)
OPpragma pragma(1)     PR_LINENUMBER Line(22)
OPpragma pragma(35)    PR_WARNING 10:OFF; 16:ERR; 72:ERR; 86:OFF;
93:OFF; 120:OFF; 205:OFF; 206:OFF; 217:OFF; 228:OFF; 231:OFF; 246:OFF;
OPblock
OPblock
OPname(?proc@@YAXXZ) symbol(0x288)
OPentry
OPeolist
***** function entry says proc has no parameters *****

**** ctor call for class1 *****
OPblock
OPpragma pragma(1)     PR_LINENUMBER Line(23)
OPname(??0class1@@QAE@XZ) symbol(0x25f)
OPname(obj1) symbol(0x28a)
OPconstant integer size(4) align(3) constant(0|0x0)
OPfield address size(4) align(3)
OPconstant integer size(4) align(3) constant(0|0x0)
OPfield address size(4) align(3) Const
OPextract address size(4) align(3) Const
OPmfunc address size(4) align(3)
OPfunction address size(4) align(3) Const functype(20)
OPeolist
*****

*** dtor call for class1 *****
OPname(??1class1@@QAE@XZ) symbol(0x260)
OPname(obj1) symbol(0x28a)
OPconstant integer size(4) align(3) constant(0|0x0)
OPfield address size(4) align(3)
OPconstant integer size(4) align(3) constant(0|0x0)
OPfield address size(4) align(3) Const
OPextract address size(4) align(3) Const
OPmfunc address size(4) align(3)
OPfunction void size(0) align(1) functype(20)
OPeolist
*****
```

3810

3820

## FIG. 38B

OPpushstate address size(4) align(3) EH Flags 0x00000011

3830

OPexpression

OPpragma pragma(1) PR\_LINENUMBER Line(24)

OPname(?foo@class1@@QAEXXZ) symbol(0x261)

OPname(obj1) symbol(0x28a)

OPconstant integer size(4) align(3) constant(0|0x0)

OPfield address size(4) align(3)

OPconstant integer size(4) align(3) constant(0|0x0)

OPfield address size(4) align(3) Const

OPextract address size(4) align(3) Const

OPmfunc address size(4) align(3)

OPfunction void size(0) align(1) functype(20)

OPeolist

OPexpression

OPpragma pragma(1) PR\_LINENUMBER Line(25)

OPname(??0class2@@QAE@XZ) symbol(0x274)

OPname(obj2) symbol(0x28b)

OPconstant integer size(4) align(3) constant(0|0x0)

OPfield address size(4) align(3)

OPconstant integer size(4) align(3) constant(0|0x0)

OPfield address size(4) align(3) Const

OPextract address size(4) align(3) Const

OPmfunc address size(4) align(3)

OPfunction address size(4) align(3) Const functype(20)

OPeolist

OPname(??1class2@@QAE@XZ) symbol(0x275)

OPname(obj2) symbol(0x28b)

OPconstant integer size(4) align(3) constant(0|0x0)

OPfield address size(4) align(3)

OPconstant integer size(4) align(3) constant(0|0x0)

OPfield address size(4) align(3) Const

OPextract address size(4) align(3) Const

OPmfunc address size(4) align(3)

OPfunction void size(0) align(1) functype(20)

OPeolist



## FIG. 38C

OPpushstate address size(4) align(3) EH Flags 0x00000011  
OPexpression  
OPpragma pragma(1) PR\_LINENUMBER Line(26)  
OPname(?bar@class2@@QAEXXZ) symbol(0x276)  
OPname(obj2) symbol(0x28b)  
OPconstant integer size(4) align(3) constant(0|0x0)  
OPfield address size(4) align(3)  
OPconstant integer size(4) align(3) constant(0|0x0)  
OPfield address size(4) align(3) Const  
OPextract address size(4) align(3) Const  
OPmfunc address size(4) align(3)  
OPfunction void size(0) align(1) functype(20)  
OPeolist  
OPexpression  
OPpragma pragma(1) PR\_LINENUMBER Line(27)  
OPdtoraction cnt(2) EH Flags 0x00000031  
OPexpression  
OPgoto symbol(0x289)  
OPendblock icon(2)  
OPlabel symbol(0x289)  
OPexit  
OPendblock icon(1)  
OPendblock icon(0)

3840

FIG. 39

